IML Project Report

Heikki Nenonen, Arttu Koskinen, Niko Petjakko

2022-12-10

Currently the report consists of the code for everything we've done, but lacks verbal explanations, which we're hoping to add in for the final report.

Todo

- ~~dummy classifier~~
- ~~class4 -> event/nonevent, week1 exc?~~
- ~~drop partlybad, pelkkää FALSEa~~
- **varianssit mukana/ei mukana? ei one hot -> yksinkertaistaa liikaa ja tarkoitettu kategoriseen dataan**
- ~~date? paljon informaatiota, mutta halutaanko muuttujaksi <- opeta 2000-2008, testaa 2009-2011 / kysy slack test_hidden ei - - date, jätetäänkö pois? good riddance!~~
- ~~train, test, cv-10?~~
- ~~itse logisticregression, week2 exc1 <- lasso/ridge~~
- ~~accuracy, perplexity, week2 exc1~~
- accuracy of our accuracy? <– malli train+test, vähän parempi kuin pelkkä train?
- ~~class4 -> nonevent/1a/1b/II/ = 0,1,2,3~~
- ~~try normalising data - google if needed for RF~~ needed for PCA
- try PCA (not implemented yet)
- multiclass classifier tod.näkösyydet eivät näy atm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model

npf_test = pd.read_csv("initial_data/npf_test_hidden.csv")
npf_train = pd.read_csv("initial_data/npf_train.csv")
```

```
npf_train_test = npf_train.set_index("date")
npf_train_test = npf_train_test.drop(['id', 'partlybad'], axis=1)

class2 = np.array(["nonevent", "event"])
class2 = class2[(npf_train_test["class4"]!="nonevent").astype(int)]
#class2 = class2.apply(lambda x: 1 if "event" else 0)
npf_train_test.insert(loc=0, column="class2", value=class2)
```

```
npf_train_test["class2"].replace(["event", "nonevent"],[1,0], inplace=True)
npf_train_test["class4"].replace(["nonevent", "Ia", "Ib", "II"],[0, 1, 2, 3], inplace=True)

#DROPS STDS
npf_train_test = npf_train_test.filter(regex='mean|class4|class2')
```

First five columns and rows

```
knitr::kable(head(py$npf_train_test[,1:5]), row.names = TRUE, digits = 2)
```

	class2	class4	CO2168.mean	CO2336.mean	CO242.mean
2000-01-17	1	2	368.77	368.67	369.37
2000-02-28	0	0	378.20	378.08	378.67
2000-03-24	1	2	373.04	372.93	373.57
2000-03-30	1	3	375.64	375.55	376.05
2000-04-04	0	0	377.66	377.61	378.12
2000-04-07	1	1	373.87	373.79	374.36

```
#@ignore_warnings(category=ConvergenceWarning)
def loss(X_tr, y_tr, X_te, y_te, m):
    return mean_squared_error(y_te, m.fit(X_tr, y_tr).predict(X_te), squared=False)


def accuracy(X_tr, y_tr, X_te, y_te, m):
    return accuracy_score(y_te, m.fit(X_tr, y_tr).predict(X_te))

#def perplexity(p, y_test):
#    return np.exp(-np.mean(np.log(y_test*p + (1 - y_test) * (1 - p))))

#perplexity = lambda p: np.exp(-np.mean(np.log(y_test*p + (1 - y_test) * (1 - p))))
```

```
def magic(models, classtype):

  X = npf_train_test.drop(["class2", "class4"], axis=1, inplace=False)
  y = npf_train_test[classtype]
  X_train, X_test, y_train, y_test = train_test_split(
      X, y, train_size=0.8, random_state=40, shuffle=True, stratify=y
  )


  res = pd.DataFrame(index=models)
  # Loss on training data, for model trained on training data:
  res["train_loss"] = [loss(X_train, y_train, X_train, y_train, m) for m in models]
  # Cross-validation loss:
  res["cv_loss"] = [
      -cross_val_score(
          m, X_train, y_train, cv=10, scoring="neg_root_mean_squared_error"
      ).mean()
      for m in models
  ]
  # Los on test data, for model trained on training data:
```

```
  res["test_loss"] = [loss(X_train, y_train, X_test, y_test, m) for m in models]
  res["test_accuracy"] = [accuracy(X_train, y_train, X_test, y_test, m) for m in models]

  perplexity = lambda p: np.exp(-np.mean(np.log(y_test*p + (1 - y_test) * (1 - p))))

  #temporary solution since sum is weird with perplexity, KEEP SVM LAST!!
  list = [perplexity(m.predict_proba(X_test)[:,1]) for m in models[0:-1]]
  res["test_perplexity1"]= np.append(list, 0)

  return res
```

```
test = results_class2.reset_index()
```

```
knitr::kable(py$results_class2, row.names = TRUE, digits = 2)
```

	train_loss	cv_loss	test_loss	test_accuracy	test_perplexity1
1	0.71	0.72	0.71	0.49	2.00
2	0.35	0.38	0.36	0.87	1.46
3	0.37	0.37	0.43	0.82	1.48
4	0.37	0.37	0.43	0.82	1.48
5	0.36	0.38	0.36	0.87	1.45
6	0.37	0.37	0.43	0.82	1.48
7	0.40	0.39	0.40	0.84	Inf
8	0.19	0.38	0.34	0.88	11.83
9	0.00	0.34	0.34	0.89	1.29
10	0.33	0.41	0.44	0.81	Inf
11	0.28	0.34	0.34	0.88	0.00

```
knitr::kable(py$results_class4, row.names = TRUE, digits = 2)
```

	train_loss	cv_loss	test_loss	test_accuracy	test_perplexity1
1	1.73	1.73	1.74	0.49	1.51
2	1.13	1.14	1.16	0.67	1.40
3	1.12	1.13	1.22	0.66	1.44
4	1.12	1.13	1.22	0.66	1.44
5	1.12	1.14	1.16	0.67	1.40
6	1.13	1.13	1.22	0.66	1.44
7	1.30	1.32	1.36	0.61	2.60
8	0.56	1.34	1.32	0.63	Inf
9	0.00	1.06	1.04	0.70	1.26
10	1.04	1.19	1.31	0.55	Inf
11	0.96	1.05	1.05	0.69	0.00

```
#as.matrix(py$test)
#knitr::kable(py$test, digits = 2)
```

```
#shows all columns
#required package tabulate
#print(res.to_markdown())
models = [
  RandomForestClassifier(criterion='gini'),
```

```
  RandomForestClassifier(criterion='log_loss'),
  RandomForestClassifier(criterion='entropy')]

results_class4 = magic(models, 'class4')
```

```
## /home/artkoski/.local/lib/python3.8/site-packages/pandas/core/arraylike.py:397: RuntimeWarning: inval
##   result = getattr(ufunc, method)(*inputs, **kwargs)
## /home/artkoski/.local/lib/python3.8/site-packages/pandas/core/arraylike.py:397: RuntimeWarning: inval
##   result = getattr(ufunc, method)(*inputs, **kwargs)
```

```
knitr::kable(py$results_class4, row.names = TRUE, digits = 2)
```

	train_loss	cv_loss	test_loss	test_accuracy	test_perplexity1
1	0	1.04	1.03	0.69	1.32
2	0	1.04	1.08	0.70	1.24
3	0	1.02	1.03	0.70	0.00

```
npf_test_clean = npf_test.drop(['id', 'partlybad', 'date'], axis=1)

#DROPS STDS
npf_test_clean = npf_test_clean.filter(regex='mean|class4|class2')

npf_train_events = npf_train_test[npf_train_test['class2']==1]
X = npf_train_events.drop(columns=['class2', 'class4'])
y = npf_train_events['class4']
rfc = RandomForestClassifier(criterion='gini')
model = rfc.fit(X, y)


predict_x = npf_test_clean.loc[final[final[0]=='event'].index].drop(columns='class4')

#final[final[0]=='event'].index

#probas['proba'] = probas.apply(lambda row: get_predict_proba(row, model), axis=1)

predicts = pd.DataFrame(model.predict(predict_x))


predicts[0].replace([1, 2, 3],["Ia", "Ib", "II"], inplace=True)

i = 0
for index, row in final.iterrows():
  if row[0]=='event':
    #print(f'i: {i}, predict: {predicts.iloc[i,0]}')
    #row[0] = predicts.iloc[i,0]
    final.at[index, 0] = predicts.iloc[i,0]
    i += 1

#final = predicts.merge(probas['proba'].to_frame(), left_index=True, right_index=True)
#final[0].replace([0, 1, 2, 3],["nonevent", "Ia", "Ib", "II"], inplace=True)
```

```
row0 = pd.DataFrame({0: 0.9, 'proba':''}, index =[0])
row1 = pd.DataFrame({0: 'class4', 'proba':'p'}, index =[0])
merged = pd.concat([row1, final])
merged = pd.concat([row0, merged])

merged.to_csv('answers.csv', index=False, header=False)
```